Claims

1. (Previously Presented) A method of generating an extended version of software written in an object-oriented programming language which provides for object classes via a plurality of extensions to the software, the method comprising:

receiving invocations of a plurality of software development scenario class extension sets comprising extensions for respective software development scenarios to be implemented by the extended version of the software, wherein the respective software development scenarios specify a plurality of target processor architectures and a plurality of compilation scenarios; and

extending one or more classes of the software as indicated by the extensions, wherein the extending comprises:

processing classes of the software indicated as being statically extensible together with their corresponding extensions to generate the extended version of the software, wherein the processing comprises:

using an object description language pre-processor to generate a header file with a source code representation of an extended class comprising the classes of the software and their corresponding extensions; and

compiling the header file to generate the extended version of the software.

2. (Original) One or more computer-readable media having computer-executable instructions for performing the method of claim 1.

3. (Canceled)

4. (Original) The method of claim 1 wherein the receiving for at least one of the extension sets occurs at runtime of the software.

5. (Original) The method of claim 1 wherein the extending comprises outputting source code for the extensions.

6. (Original) The method of claim 1 wherein the extension sets comprise:

an extension set for implementing a target architecture.

7. (Original) The method of claim 6 wherein the extension sets comprise:

an extension set for implementing a compilation scenario.

8. (Original) The method of claim 6 wherein the extension sets comprise an extension set for implementing a managed code scenario, and the method further comprises:

based on the receiving, extending the classes to provide managed code functionality.

9. (Original) The method of claim 1 wherein an invocation for at least one of the extension sets indicates is received at runtime.

10. (Original) The method of claim 1 wherein at least one of the extensions indicates an additional class member for at least one of the object classes of the software.

11. (Previously Presented) A method of extending software written in a programming language by adding extensions to a core version of the software to generate an extended version of the software, the method comprising:

receiving a configuration of the extended version of software;

receiving in an object description language definitions of extensions to classes of the core version of the software according to the configuration of the extended version of the software, wherein the classes of the core version of the software are indicated in the object description language as statically extensible prior to compile time or dynamically extensible at runtime, wherein the classes of the core version of the software indicated as being statically extensible are processed together with their corresponding extensions, and wherein the classes of the core version of the software indicated as being dynamically extensible are processed separate from their corresponding extensions; and

processing the classes of the core version of the software and the extensions to generate the extended version of the software, wherein processing the classes of the core version of the software together with their corresponding extensions comprises:

using an object description language pre-processor to generate a header file with a source code representation of an extended class comprising the classes of the core version of the software and their corresponding extensions; and

compiling the header file to generate the extended version of the software.

12-14. (Canceled)

15. (Previously Presented) The method of claim 11, wherein processing the classes of the core version of the software separate from their corresponding extensions comprises:

using an object description language pre-processor for generating a header file comprising a source code version of the classes of the core version of the software; and

compiling the header file to generate a computer executable version of the classes of the core version of the software.

16. (Previously Presented) The method of claim 11, wherein processing the classes of the core version of the software separate from their corresponding extensions comprises:

using an object description language pre-processor for generating a header file comprising a source code version of the extensions to classes of the core version of the software; and

compiling the header file to generate a computer executable version of the extensions to classes of the core version of the software.

17. (Original) The method of claim 16, further comprising processing the computer executable version of the extensions to classes of the core version of the software to generate extended classes by linking the classes of the core version of the software to their respective extensions at runtime.

18. (Original) The method of claim 11, wherein the definition of the classes of the core version of the software comprises one or more extension points for indicating points within code related to the core version of the software where code related to the extensions to classes of the core version of the software is injected.

19. (Original) The method of claim 11, wherein the core version of the software is an extensible core compiler framework and the extended version of the software is a customized compiler and receiving the configuration of the extended version of the software comprises obtaining a compiler type.

20. (Original) The method of claim 19, wherein the compiler type is selected from a group consisting of a JIT compiler, a Pre-JIT compiler and a Native Optimizing Compiler.

21. (Original) The method of claim 11, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and receiving the configuration of the extended version of software comprises obtaining a target type.

22. (Original) The method of claim 11, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and receiving the configuration of the extended version of software comprises obtaining a feature type.

23. (Previously Presented) A system for extending software by adding extensions to a core version of the software to generate an extended version of the software, the system comprising:

a computer processor for executing an object description language pre-processor operable for receiving extensions to classes of the core version of the software in an object description language and generating a source code version of the extensions to classes of the core version of the software, wherein the classes of the core version of the software are indicated in the object description language as being dynamically extensible at runtime or statically extensible prior to compile time, wherein the object description language pre-processor is programmed for processing the classes of the core version of the software indicated as being dynamically extensible separate from their corresponding extensions, and wherein the object description language pre-processor is operable for processing the classes of the core version of the software indicated as being statically extensible together with their corresponding extensions to generate a header file with a source code version of extended classes comprising the source code version of the classes of the core version of the software and their corresponding extensions; and

a compiler for compiling the header file with the source code version of the extended classes to generate a computer-executable version of the extended classes to be used for generating the extended version of the software.

24. (Canceled)

25. (Canceled)

26. (Original) The system of claim 23, wherein the classes of the core version of the software further comprise extension points for indicating locations within code related to the core version of the software where code related to the extensions are injected.

27. (Canceled)

28. (Previously Presented) The system of claim 23, further comprising a compiler for compiling the source code version of the extensions to the classes of the core version of the software to generate a computer-executable version of the extensions.

29. (Original) The system of claim 28, further comprising a computer processor for executing the computer-executable version of the extensions to generate an extended version of the software at runtime by linking the classes of the core version of the software to their corresponding extensions.

30. (Original) The system of claim 29, wherein the processor executes the computer-executable version of the extensions by invoking a computer-executable version of the core version of the software and injecting the extensions into code related to the core version of the software at runtime.

31. (Original) The system of claim 23, wherein the extensions correspond to a configuration of the extended version of the software.

32. (Original) The system of claim 31, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and the configuration of the extended version of software comprises a target type.

33. (Original) The system of claim 31, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and the configuration of the extended version of software comprises a compiler type.

34. (Original) The system of claim 31, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and the configuration of the extended version of software comprises a feature type.

35. (Previously Presented) A computer readable storage medium having stored thereon class declarations of classes of a core version of a software to be extended by extension declarations for extending the core version of the software to generate an extended version of the software, the class declaration of the core version of the software comprising:

one or more core class members; and

one or more extensibility attributes of the core classes, wherein the extensibility attribute indicates that the core classes are either statically extensible prior to compile time or dynamically extensible at runtime, wherein processing the classes of the core version of the software together with their corresponding extension declarations comprises:

using an object description language pre-processor to generate a header file with a source code representation of an extended class comprising the classes of the core version of the software and their corresponding extension declarations; and

compiling the header file to generate the extended version of the software;

wherein the extension declarations correspond to a particular configuration of the extended version of the software and the extension declaration further comprise one or more attribute declarations for indicating the configuration of the extended version of the software, and wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and the configuration of the extended version of software comprises choosing a target processor architecture.

36. (Original) The computer readable storage medium of claim 35, further comprising the extension declarations, wherein the extension declarations comprise one or more extension class members for extending the core version of the software.

37. (Canceled)

38. (Previously Presented) The computer readable storage medium of claim 35, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and choosing the configuration of the extended version of software comprises choosing a compiler type.

39. (Canceled)

40. (Original) The computer readable storage medium of claim 35, further comprising extension points for indicating locations within code related to the core version of the software where code related to their corresponding extensions should be injected.

41. (Previously Presented) A computer readable storage medium having stored thereon software code for a pre-processor program, wherein the pre-processor program is operable for receiving, in an object description language, classes of a core version of a software and corresponding extensions to the classes of the core version of the software to be used for extending the core version of the software, wherein the classes of the core version of the software are indicated in the object description language as being statically extensible prior to compile time or dynamically extensible at runtime, wherein the pre-processor program is further operable for using the extensions of the classes of the core version of the software received in form of the object description language to generate a source code version of the extensions to be used for linking the extensions to their corresponding classes of the core version of the software at runtime to extend the core version of the software, and wherein the pre-processor program is further operable for using the classes of the core version of the software and their corresponding extensions received in form of the object description language to generate a header file with a source code version of extended classes comprising the classes of the core version of the software and their corresponding extensions, wherein the pre-processor program is further operable for outputting the header file to a compiler for generating an extended version of the software.

42. (Canceled)

43. (Canceled)

44. **(Currently Amended)** A system for extending software by adding extensions to a core version of the software to generate an extended version of the software, the system comprising:

~~**executing an object description language pre-processor for**~~ means **for executing an object description language pre-processor** for receiving extensions to classes of the core version of the software in an object description language and generating a source code version of the extensions to classes of the core version of the software, wherein the classes of the core version of the software are indicated in the object description language as being dynamically extensible at runtime or statically extensible prior to compile time, wherein the classes of the core version of the software indicated as being statically extensible are processed together with their corresponding extensions, wherein the classes of the core version of the software indicated as being dynamically extensible are processed separate from their corresponding extensions, wherein processing the classes of the core version of the software together with their corresponding extensions comprises:

using **[[an]] the** object description language pre-processor to generate a header file with a source code representation of an extended class comprising the classes of the core version of the software and their corresponding extensions; and

compiling the header file to generate the extended version of the software; and

means for compiling the source code version of the extensions to classes of the core version of the software to be used for generating the extended version of the software.